LA-UR-21-31989
Approved for public release; distribution is unlimited.

Title: Computational Astrophysics in the Era of Technological Heterogeneity

Author(s): Dolence, Joshua C.

Intended for: Colloquium presentation

Issued: 2021-12-08

Los Alamos
NATIONAL LABORATORY

Computational Astrophysics in the Era of Technological Heterogeneity

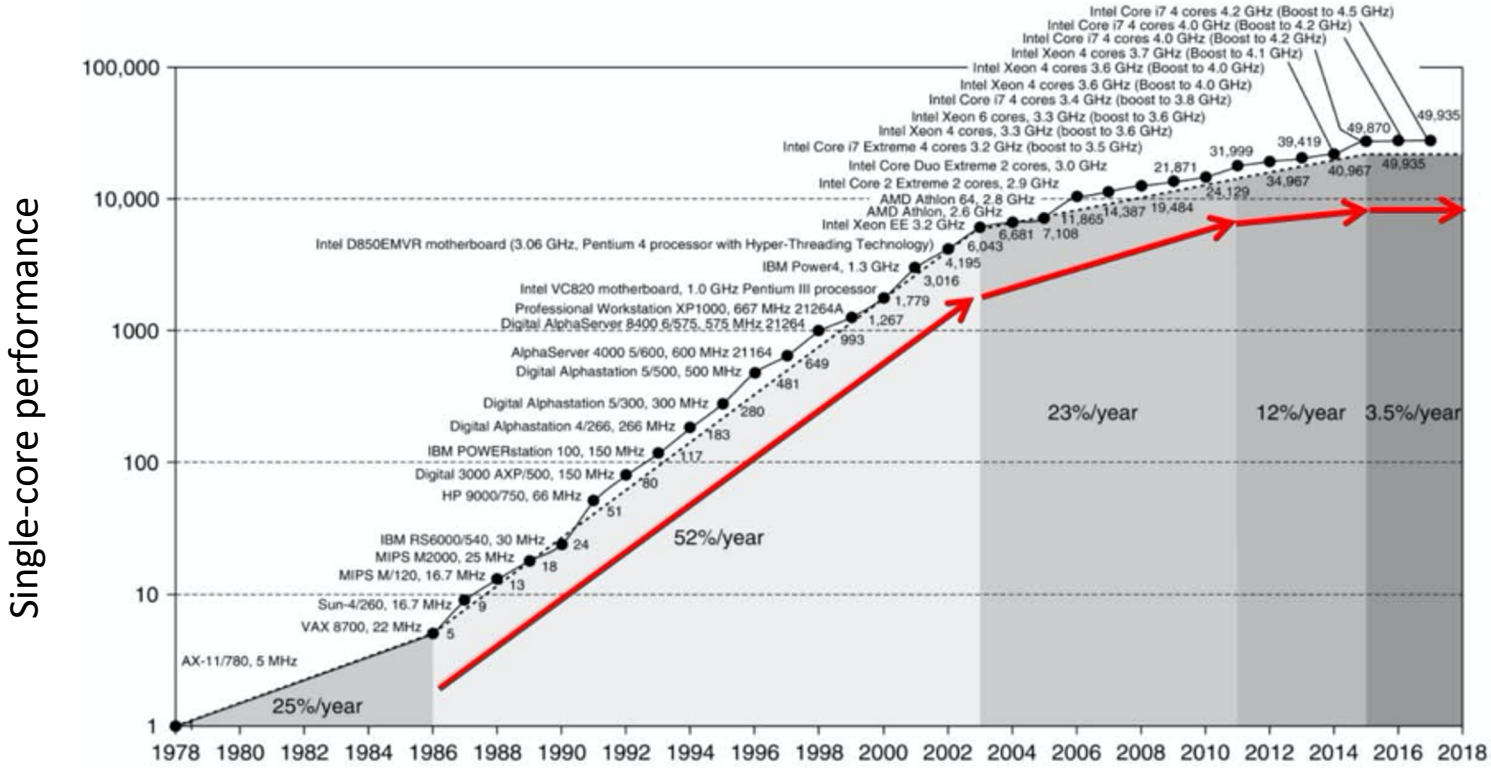
Josh Dolence
Los Alamos National Laboratory

November 30, 2021

Outline

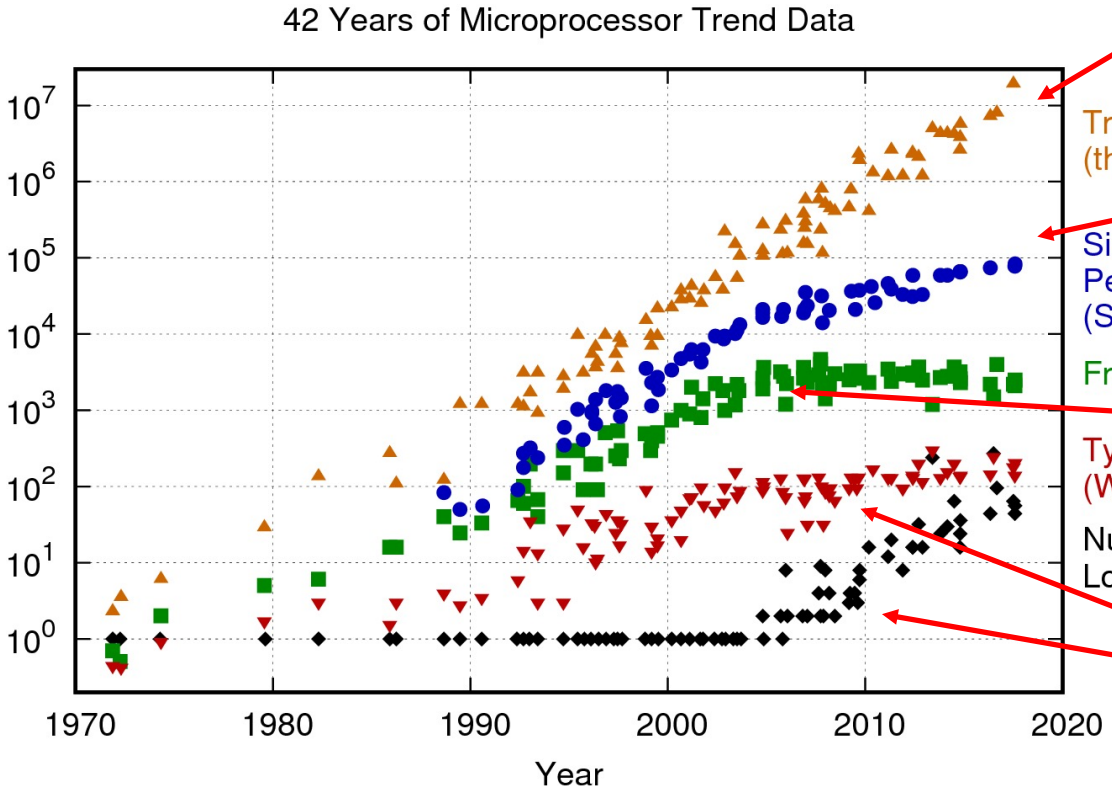
- A profound shift in computing
- Challenges on the road ahead
- Parthenon, an exemplar of a productive path for computational astrophysics
- Phoebus, an exciting new code for relativistic astrophysics

The end of an era



Hennessy & Patterson, 2018

CPUs don't have much runway



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

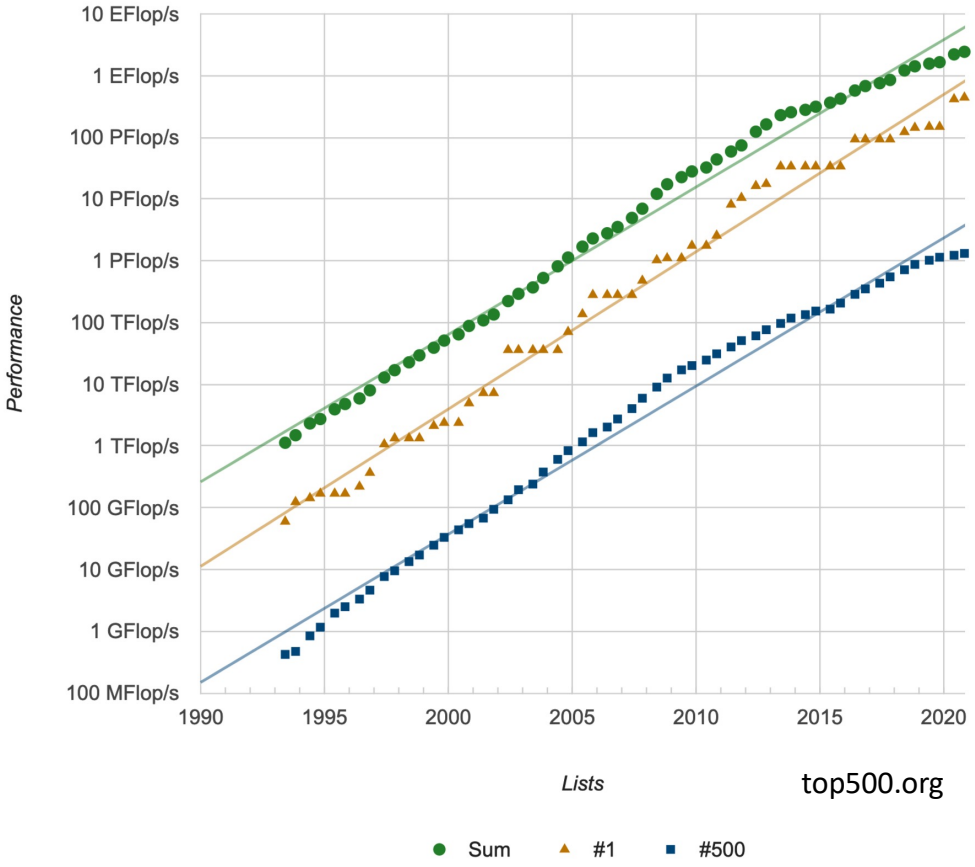
Moore's law isn't quite dead

But single-thread performance isn't reaping the benefits

The end of Dennard scaling is the end of the "easy" road

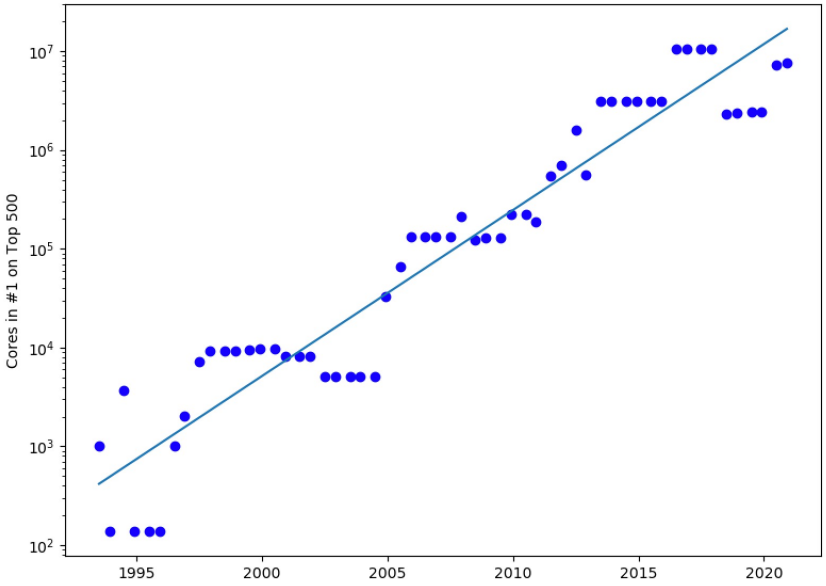
The rise of multicore, new inefficiencies, dark silicon, fixed TDP

HPC systems keep getting faster…

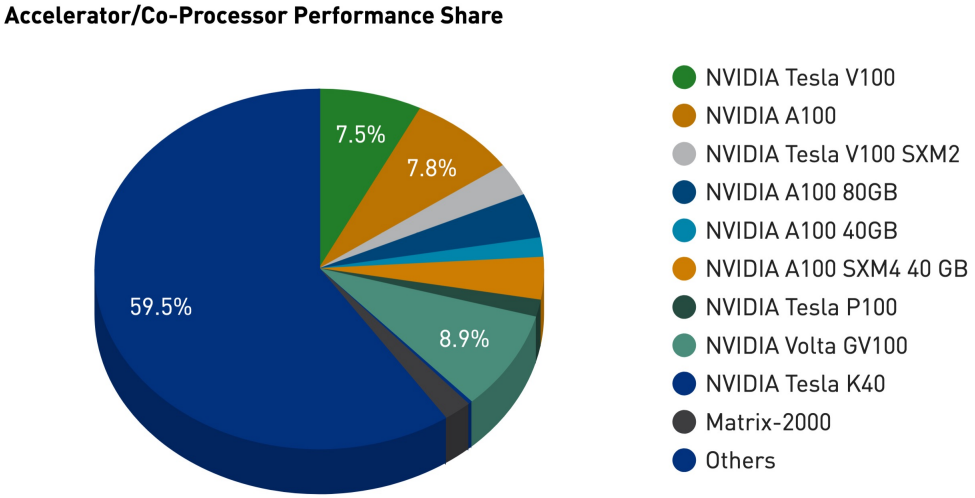


top500.org

…but that comes at a cost



Systems are getting ***really*** big!
2x every 1.8 years. 10x every 6 years.



Accelerators are increasingly common.
40% of aggregate performance in NVIDIA GPUs.

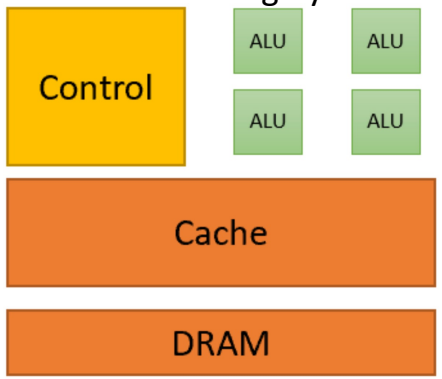
Los Alamos NATIONAL LABORATORY

Why accelerators?

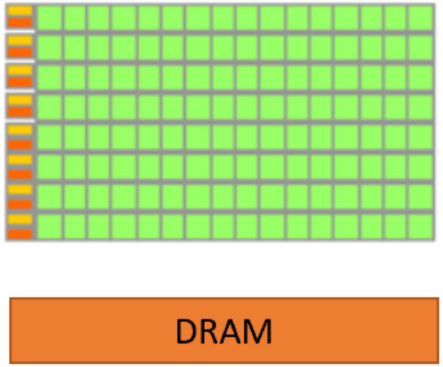
- Like everything else in computing, it's all about ***power***
- Top systems now run at 10-30 MW!
- Accelerators can deliver more flops/watt

Roughly the same number of transistors

Great at editing Microsoft Word documents



CPU



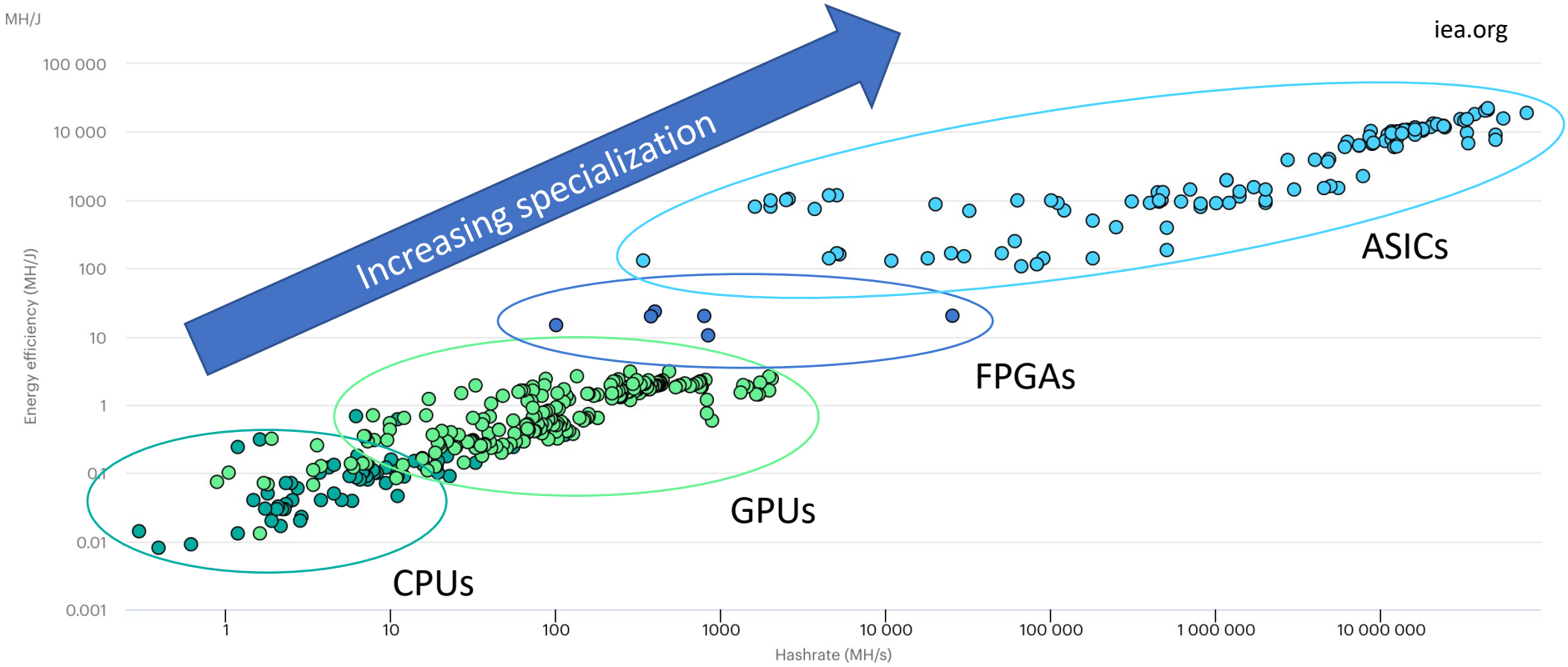
GPU

Great at doing math with dense matrices

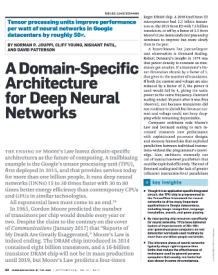
nvidia.com

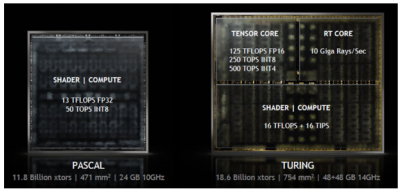
Specialization can dramatically improve efficiency

Evolution of bitcoin mining

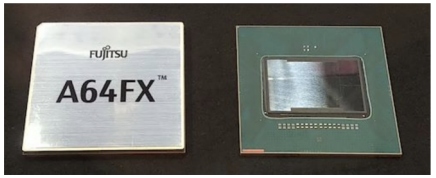


Specialized technologies are the future



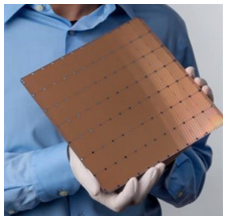


NVIDIA adding more specialization

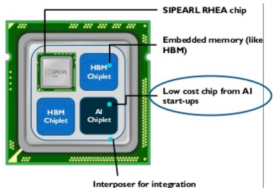


ARM on Fugaku

Microsoft project catapult

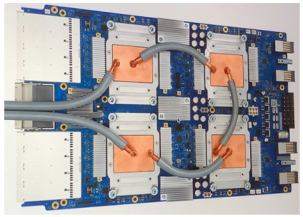


Cerebras Systems AI Wafer

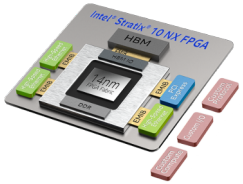


European Processor Initiative

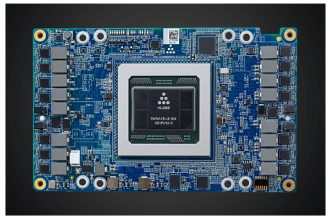
Amazon Graviton 2



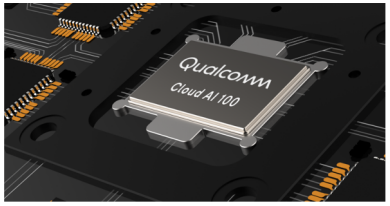
Google's Tensor Processing Unit



Intel Stratix 10 NX FPGA with integrated HBM2



Habana / Intel Gaudi AI Chip



Qualcomm Cloud AI 100

Japan charts a course for future systems

Co-Design for A64FX Manycore Processor and "Fugaku"

Mitsuhisa Sato*, Yutaka Ishikawa*, Hirofumi Tomita*, Yuetsu Kodama*, Tetsuya Odajima*, Miwako Tsuji*, Hisashi Yashiro*, Masaki Aoki†, Naoyuki Shida†, Ikuo Miyoshi†, Kouichi Hirai†, Atsushi Furuya†, Akira Asato†, Kuniki Morita†, Toshiyuki Shimizu†
* RIKEN Center for Computational Science
Email: {msato, yutaka.ishikawa, htomita, yuetsu.kodama, tetsuya.odajima, miwako.tsuji, h.yashiro}@riken.jp
† FUJITSU LIMITED, Japan
Email: {m-aoki, shidax, miyoshi.ikuo, k-hirai, furuya.atsushi, asato, morita.kuniki, t.shimizu}@fujitsu.com

Co-Design = Vertical integration: hardware architects, software stack, and application scientists in co-design

"For future systems beyond exascale, a more disruptive architecture, such as accelerators and specialized hardware, would be required."

Extreme Heterogeneity 2018:
Productive Computational Science in the Era of Extreme Heterogeneity

Report for
DOE ASCR Basic Research Needs Workshop on Extreme Heterogeneity

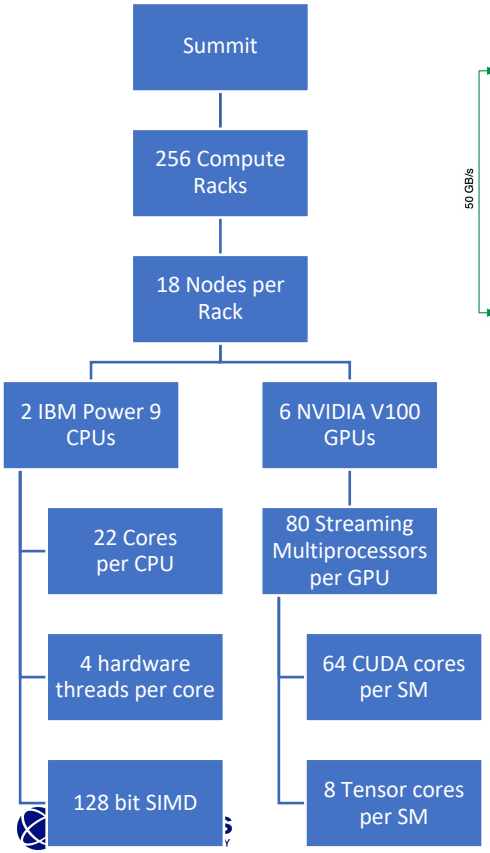
January 23–25, 2018

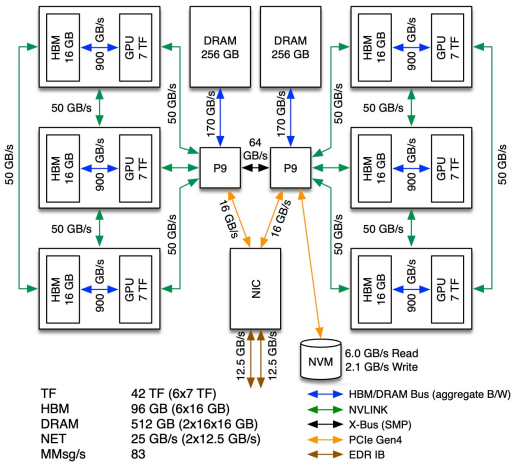
ASCR Workshop on Reimagining Codesign

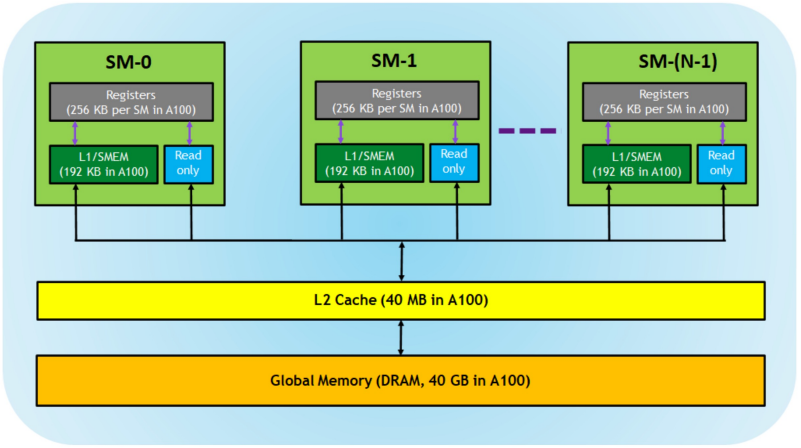
Sponsored by the U.S. Department of Energy, Office of Advanced Scientific Computing Research

March 16-18, 2021
11:00 am to 5:00 pm ET

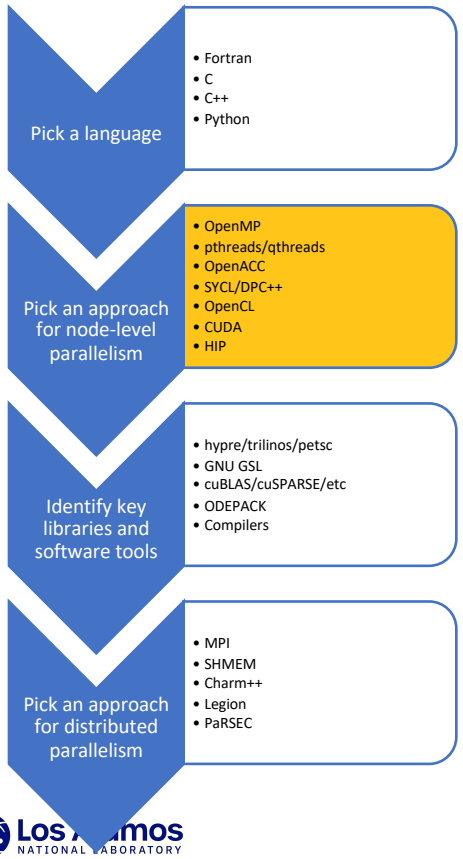
Modern systems are complex and deeply hierarchical







Writing code for modern HPC systems: the old way



Choices here constrain below

Choices here constrain architectures

Not always supported on targets

Can require careful design from the start

Astrophysics codes with CUDA for GPUs

(incomplete list)

- RamsesGPU
- Cholla
- H-AMR
- AMReX-based codes
- GAMER
- Enzo

Compiler preprocessor used to select largely distinct code paths for different architectures

Writing code for modern HPC systems: a better way

- Technological diversity has motivated the development of abstractions for *performance portability*
- Wraps CUDA, OpenMP, SYCL, etc.
- Goal is compelling performance across architectures with single source implementations

Kokkos (Sandia) provides abstractions for data and parallel execution that enable codes to run transparently across architectures

Kokkos kernels provide portable math libraries

Case-study: K-Athena achieves comparable CPU performance to Athena++ and better GPU performance than similar native CUDA codes. See Grete+20.

RAJA

Raja/CHAI provides similar capabilities. (Livermore)

Writing code for modern HPC systems: a better way

- Widely-adopted and well-supported abstractions for performance portability "solve" a major challenge with the heterogeneous HPC landscape
- But serious challenges remain
  - Optimal algorithms can differ across architectures
  - Abstractions can't change fundamental limitations of architectures – still have to have knowledge of architectures to write performant code
  - Extreme parallelism and Amdahl's law – a Summit node has ~3×10^4 CUDA cores
  - Extreme distributed scale
    - Asynchrony is a must
    - Load balancing is essential
    - I/O is a pain
    - Compute-in-network?

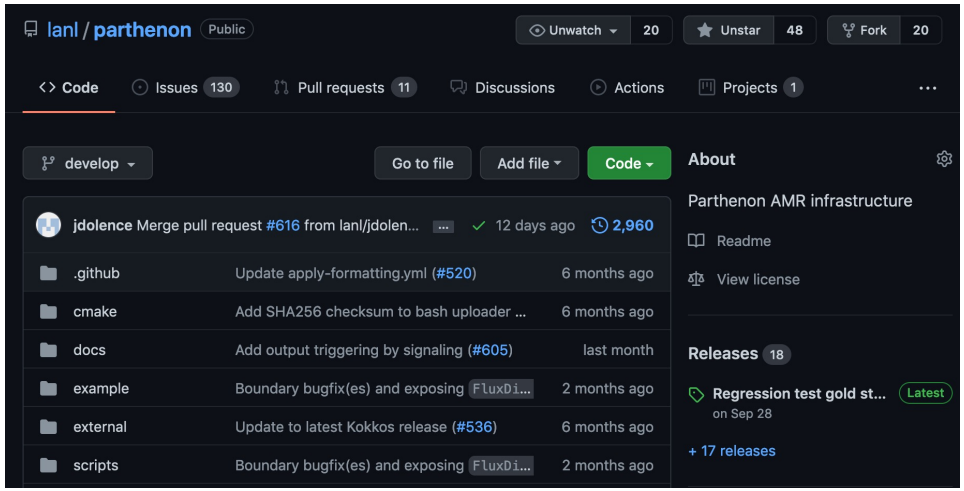
Writing code for modern HPC systems: a better way

- Widely-adopted and well-supported abstractions for performance portability "solve" a major challenge with the heterogeneous HPC landscape
- But serious challenges remain
  - Optimal algorithms can differ across architectures
  - Abstractions can't change fundamental limitations of architectures – still have to have knowledge of architectures to write performant code
  - Extreme parallelism and Amdahl's law – a Summit node has ~3×10^4 CUDA cores
  - Extreme distributed scale
    - Asynchrony is a must
    - Load balancing is essential
    - I/O is a pain
    - Compute-in-network?

Solutions to these problems can be shared across applications of a similar flavor.
Domain specific abstractions.

Parthenon: a performance portable, physics agnostic framework for block-structured AMR applications

- What is it? A “framework” designed to enable straightforward development of fast, portable, scalable physics codes built on block-structured AMR
- Layering on domain specific data types and loop abstractions can make it straightforward to write consistently performant code
- What does it provide?
  - Mesh management w/AMR
  - Load balancing
  - Communication
  - I/O
  - Data structures to store and organize mesh-based data and particle data
  - Memory management
  - Task-based parallelism
  - Abstractions to flexibly represent parallel loops
  - Adjustable parameters to adapt to different architectures



github.com/lanl/parthenon

Parthenon basics

- Downstream applications define a driver that expresses the basic control flow
- Represented as a list of *tasks* and their *dependencies*

```
TaskList tl;

TaskID none(0);
auto task_a = tl.AddTask(none, some_package::compute_a, arg1, arg2, arg3, …);
auto task_b = tl.AddTask(none, other_package::compute_b, argA, argB, …);
auto task_c = tl.AddTask(task_a|task_b, third_package::compute_c, …);

…

// task list gets executed, perhaps concurrently with other task lists
```

Task dependency

A task with one or more kernels

Arguments to task function

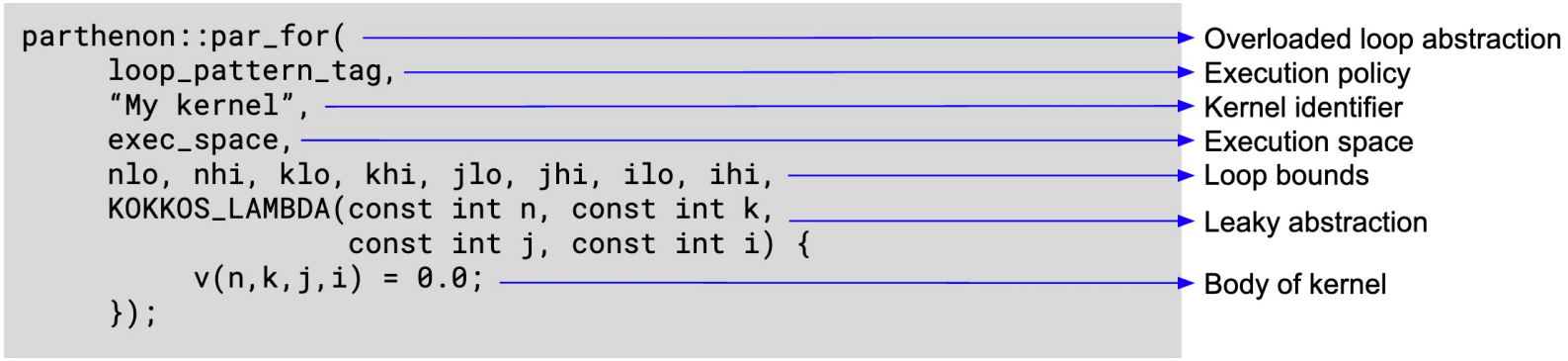
These tasks can execute concurrently

Depends on task_a and task_b

- Task parallelism enables asynchrony and concurrency

Parthenon basics

- Typical loops are replaced with Parthenon-provided constructs



- Behind the scenes, changing tags can transform this into a Kokkos loop with different execution policies, basic C++ for loops, and other approaches are being implemented

Parthenon basics

- Applications describe the fields on the mesh and Parthenon manages the memory and provides simple accessors

```
Metadata m({Metadata::Cell, Metadata::Independent, Metadata::Intensive,
            Metadata::Conserved, Metadata::WithFluxes});
hydro->AddField("density", m);
```

```
auto my_var = mb_data->Get("my favorite variable").data; // outside kernel
…
my_var(k,j,i) = 0.0;  // inside a kernel
```

Access a field on a block of cells

```
auto my_pack = mb_data->PackVariables({"my var1", "my var2", …}); // outside kernel
…
my_pack(var_index,k,j,i) = 0.0;  // inside a kernel
```

Access several fields on a block

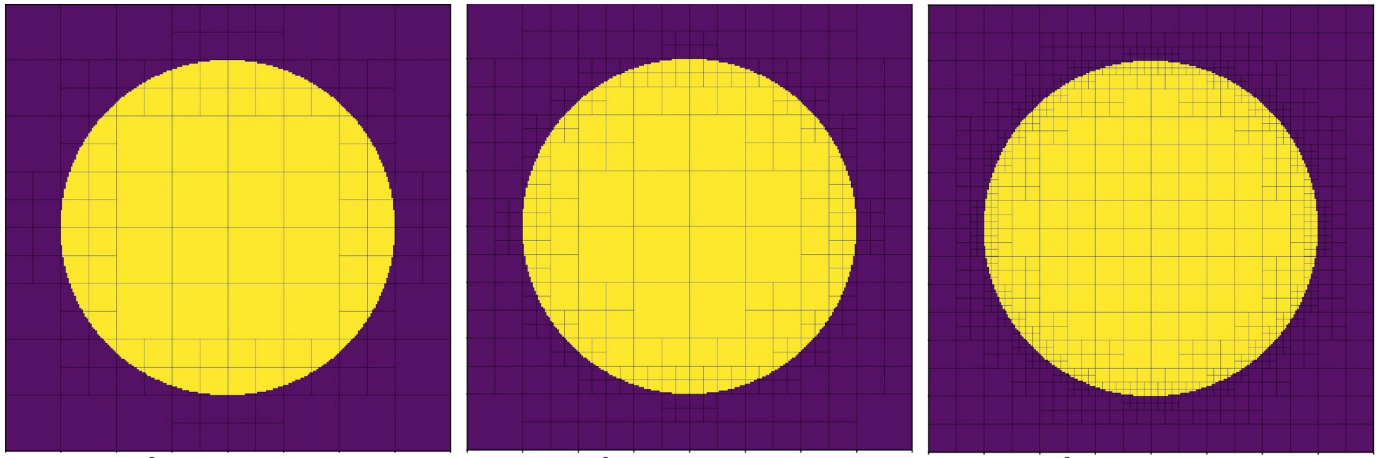
```
auto my_pack = m_data->PackVariables({"my var1", "my var2", …}); // outside kernel
…
my_pack(block_index,var_index,k,j,i) = 0.0;  // inside a kernel
```

Access fields on several blocks

Data packaging provides opportunities to control task granularity

Parthenon basics

- Includes useful examples that serve as starting points for downstream apps



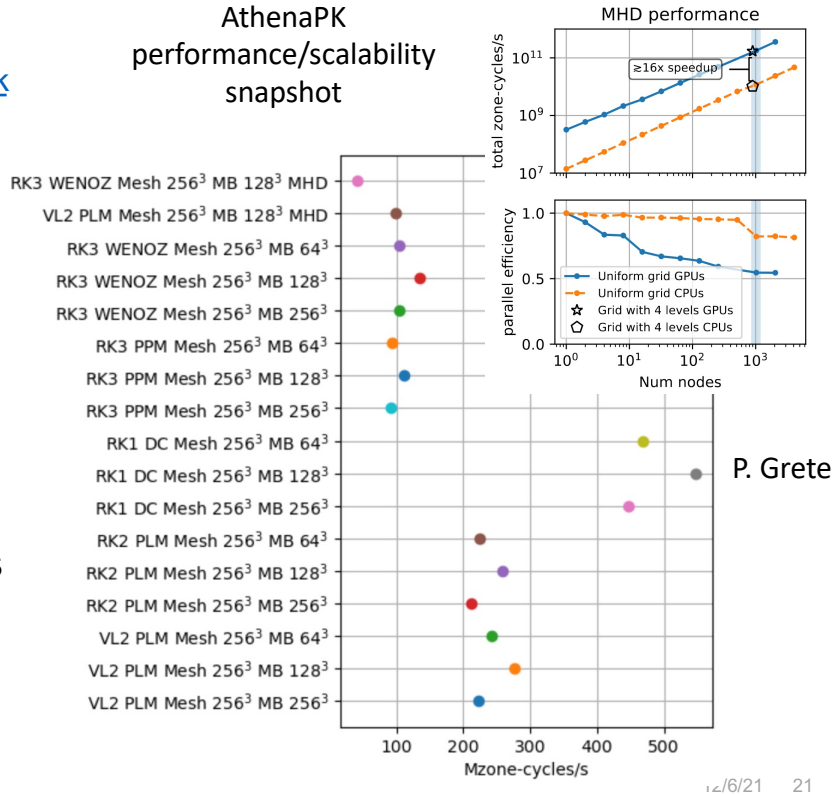
16^2 cells per block | 8^2 cells per block | 4^2 cells per block

- Calculate π with adaptive mesh refinement!
- Basic hydro code with AMR that's *very* scalable and performant on CPUs and GPUs can be written in ~a day. I've done it!

Multiple codes are already based on Parthenon

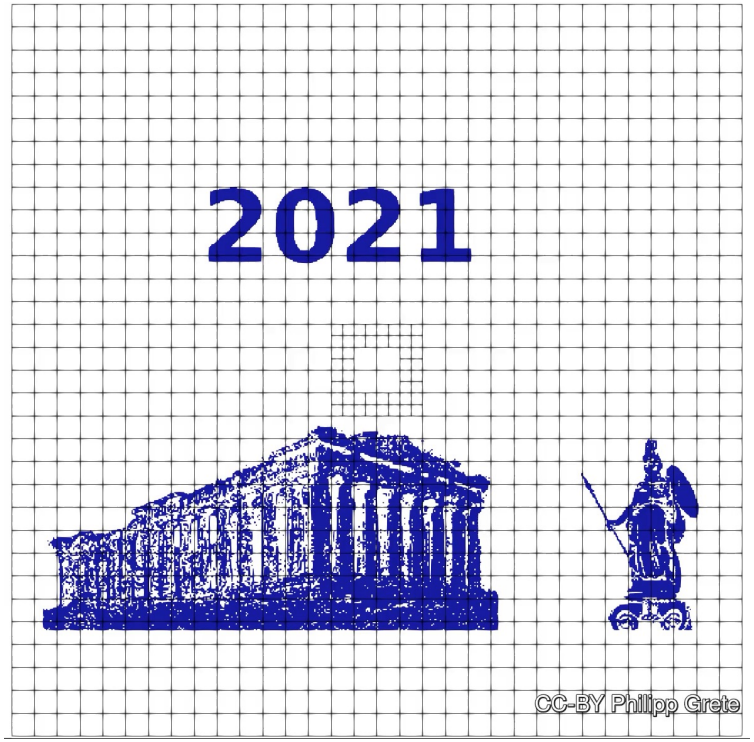
- AthenaPK
  https://gitlab.com/theias/hpc/jmstone/athena-parthenon/athenapk
  - Hydro, MHD, thermal conduction
- KHARMA https://github.com/AFD-Illinois/kharma
  - GRMHD. Ask B. Prather and C. Gammie!
- Phoebus https://github.com/lanl/phoebus
  - More on this later…
- RIOT
  - Multi-material hydro, radiation diffusion, subgrid turbulence, reactions, high explosives
- Zapp
  - Kinetic physics

AthenaPK performance/scalability snapshot



P. Grete

A film by Phil Grete…



Phoebus: a new open source code for relativistic astro

Phoebus = (10^) PHifty One Ergs Blows Up a Star

Target Applications

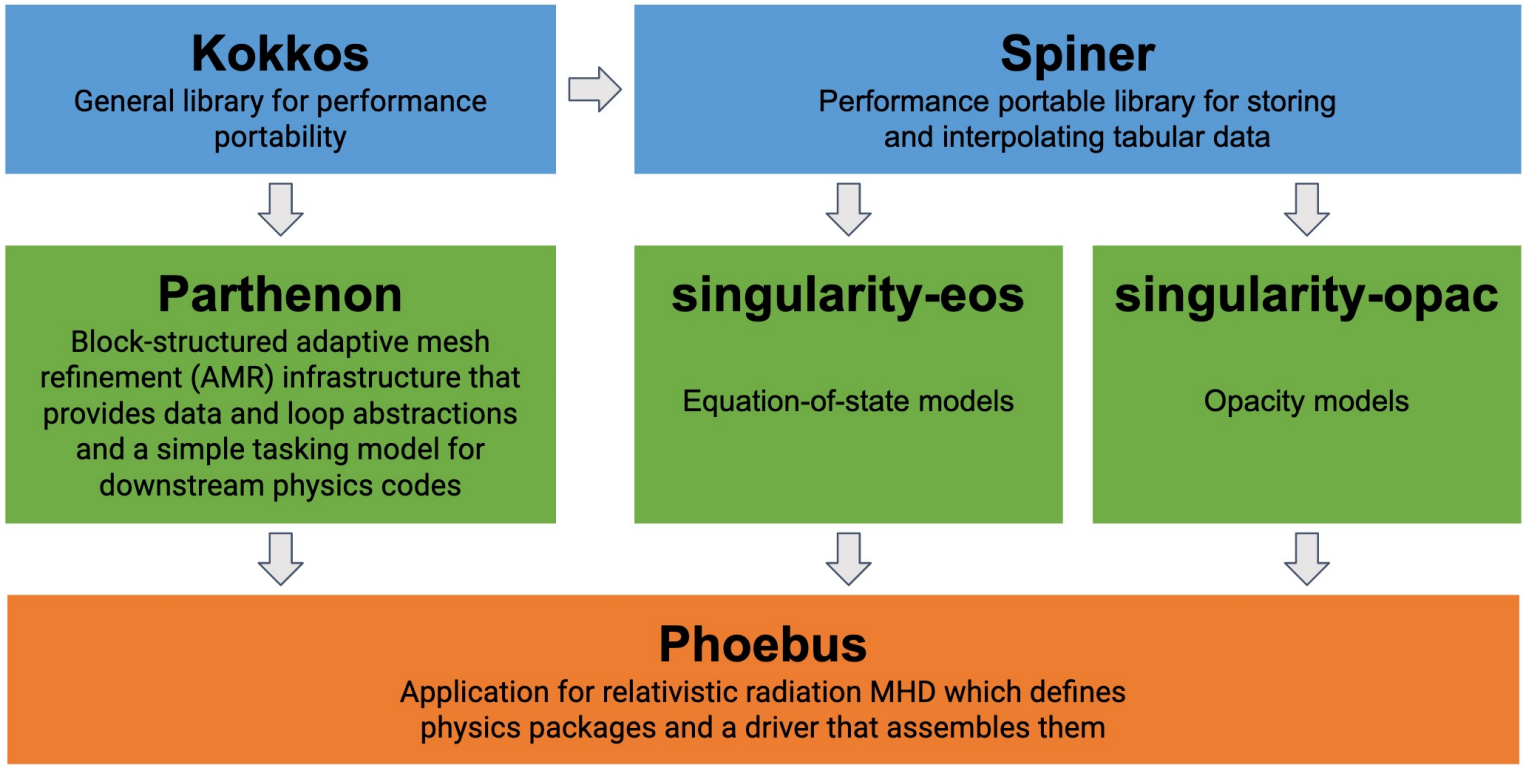
- Accreting single and binary black holes
- Core-collapse supernovae
- Compact object mergers

Physics

- Relativistic hydro/MHD
- Stationary, time-dependent, and dynamical spacetimes
- Radiation transport
- "Realistic" microphysics

- *Almost* open source – just need to click the button
- Very active development to build out all the necessary capabilities – not everything is available yet
- Core team: Jonah Miller, Ben Ryan, Luke Roberts, Josh Dolence
- Excited to welcome new collaborators!

Phoebus is built on portable, open source components



MOCMC: a new method for relativistic transport

(Ryan & Dolence, 2020)

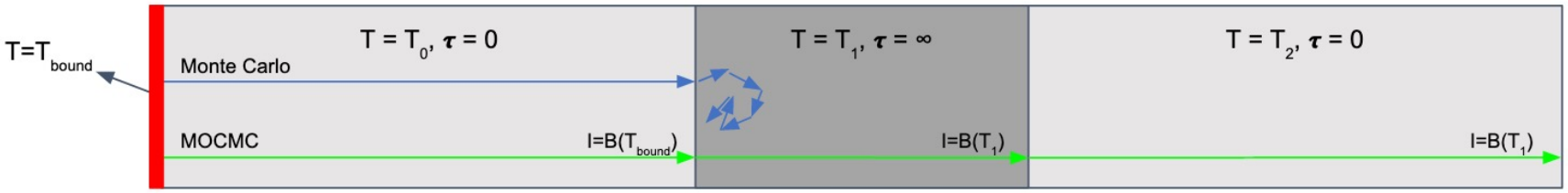
Hybrid method evolves a gray moment system closed with spectral and angular information from samples of the full distribution function.

Moment Sector

- Represents conservation of energy-momentum
- Mediates interactions of radiation and matter via four-force
- Transport of energy-momentum determined by appropriately averaged closure

Sample Sector

- Lagrangian "particles" that hold complete species/spectral information at a point in space and in a particular direction
- Samples glued to geodesics
- Sample intensities updated via transport equation



Conclusions

- The HPC landscape is changing and we're headed toward an era where compute specialization will be prevalent
- There are opportunities for co-design that can influence this future
- Technological heterogeneity will be a **major** challenge unless we shift our approach to developing the computational tools for astrophysics
- Parthenon provides convenient functionality and a bright future for block structured AMR applications
- Phoebus is a new (soon-to-be) open source code for relativistic astro that promises excellent performance, portability, and unique physics capabilities